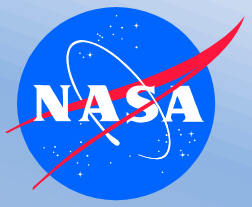
National Aeronautics
and Space Administration



Modern Microprocessor SEE Testing

Steven M. Guertin

Jet Propulsion Laboratory / California Institute of Technology

Pasadena, CA

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology,
Under contract with the National Aeronautics and Space Administration (NASA)

This work was sponsored by the NASA Electronic Parts and Packaging (NEPP) program. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

National Aeronautics
and Space Administration

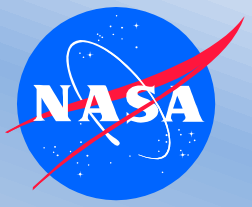
Outline

- Motivation
- Background
- Processor SEE Evaluation – old & new
- Moving Forward
- Conclusions

National Aeronautics
and Space Administration

New Processors are Here!

- Lots of new and upcoming missions are using new devices* (SOC architectures)
  - Cell phone processors/drone processors on Cubesats and Mars Helicopter
  - (see https://rotorcraft.arc.nasa.gov/Publications/files/Balaram_AIAA2018_0023.pdf)
  - RAD5545 – quad core 64-bit PowerPC being considered for several programs
  - Interest in GPUs (nVidia & AMD) for machine learning, Intel devices, Qualcomm cellphone processors, etc.
- SEE testing of these devices, especially commercial, is running into problems



What Missions Care About

- Interestingly, many assume that SEEs will result in an incorrect result.
  - Almost all modern processors have error correction.
  - They are much more likely to catch an error and throw an exception.
- Availability and reliability
  - Will it be there when they need it
- Primary event types…
  - Permanent damage, reliability impacts
  - Uncontained mistakes
  - Crashes that automatically reset
  - Crashes requiring intervention

National Aeronautics
and Space Administration

Things Changing

- New processors are different, and everything's making testing harder
  - SOCs – they're like testing an entire board
    - Isolating and collecting data from a single subsystem, and is that an appropriate test?
  - Multicore – heterogeneous processors
    - Secure boot
  - Packaging, stacked die
- One of the biggest reasons we're having trouble is …

National Aeronautics
and Space Administration

Documentation

- In the 80s and 90s, microprocessors had volumes of available documentation…
  - (Almost) All intended behaviors documented
  - Support for direct operation of hardware
- Today, most microprocessors come with only a small amount of documentation.
  - Typically a system only provides a “getting started” pamphlet and a pointer to OS instructions
- What changed?
  - Security – low level info helps hackers (keyloggers)
  - IP protection – not new… but now worse
  - Intel’s Software Developer’s Manual is ~5000 pages and is only marginally helpful for figuring out what a cache bit error machine check exception means…

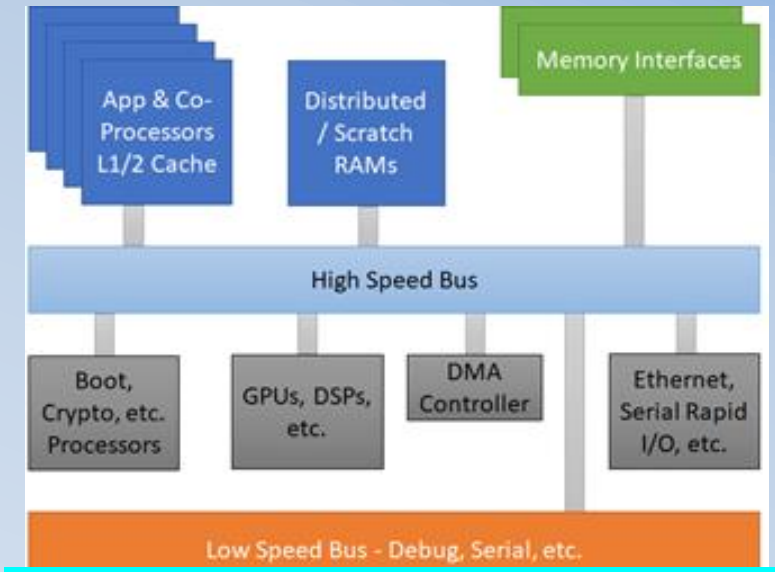
VS…

National Aeronautics and Space Administration

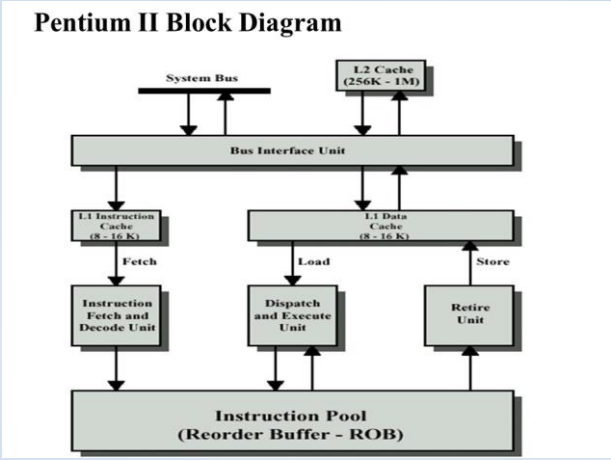
Processor Changes

- Methods that have been in place were developed against much simpler CPUs…
- Newer devices are very complex
- Even new RHBD processors are using this type of architecture…

- And, it is already old…



Modern heterogeneous microprocessor



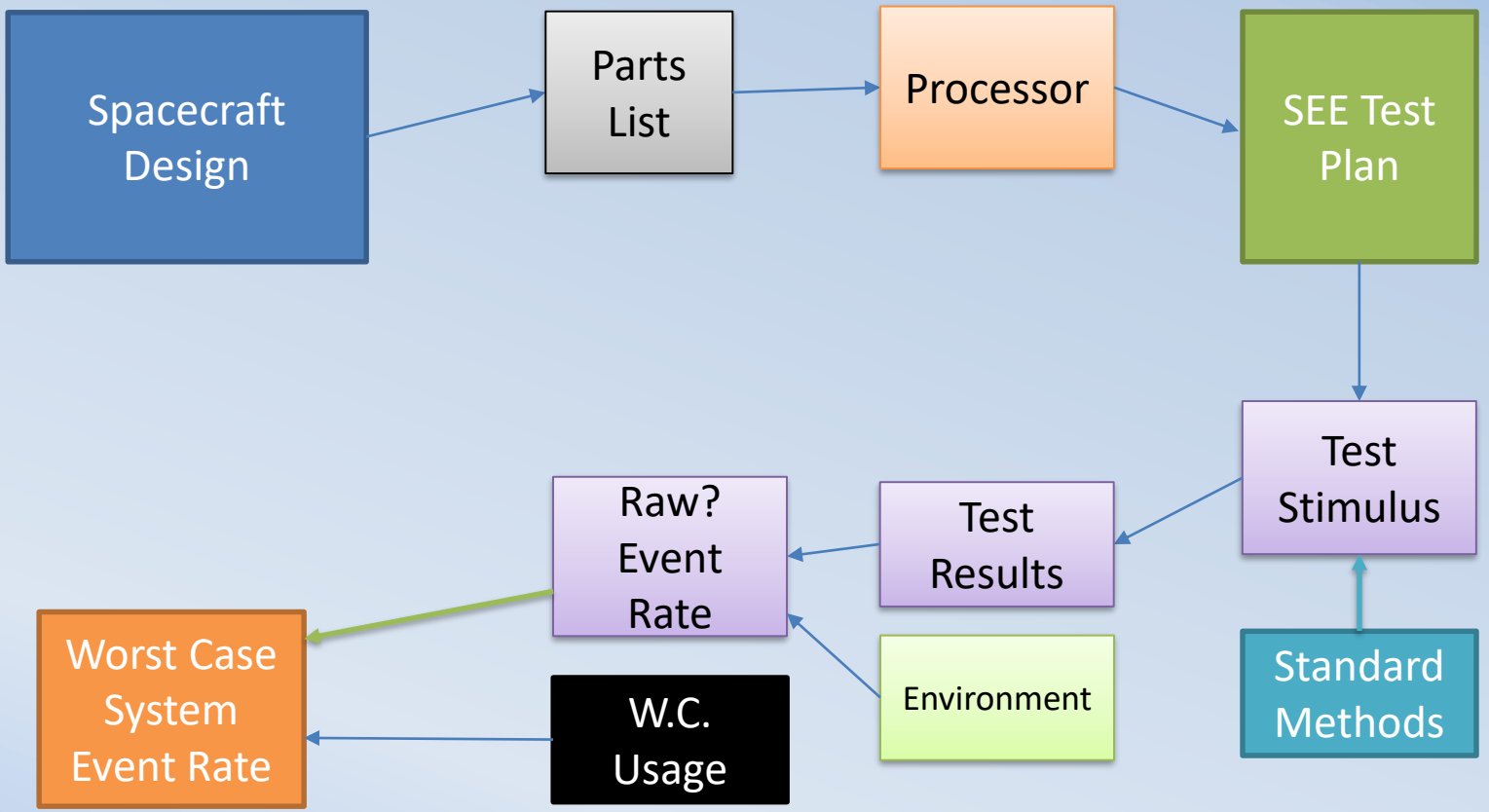
http://slideplayer.com/slide/5953241/

(Old SEE Evaluation)

- Permanent damage/reliability
  - This is a basic SEL test
- Uncontained errors
  - Evaluate the basic sensitivity of: registers, caches, operations, and scale to the maximum number that could be used **(essentially ignores SOC issues)**
- Crashes with (& without) automatic reset
  - Take basic sensitivity and scale to the total number of elements in the device – assume the majority cause reset – that is, **assume worst case**
  - With/without automatic reset is the designer's problem (unless they say, we assume all are without)

National Aeronautics
and Space Administration

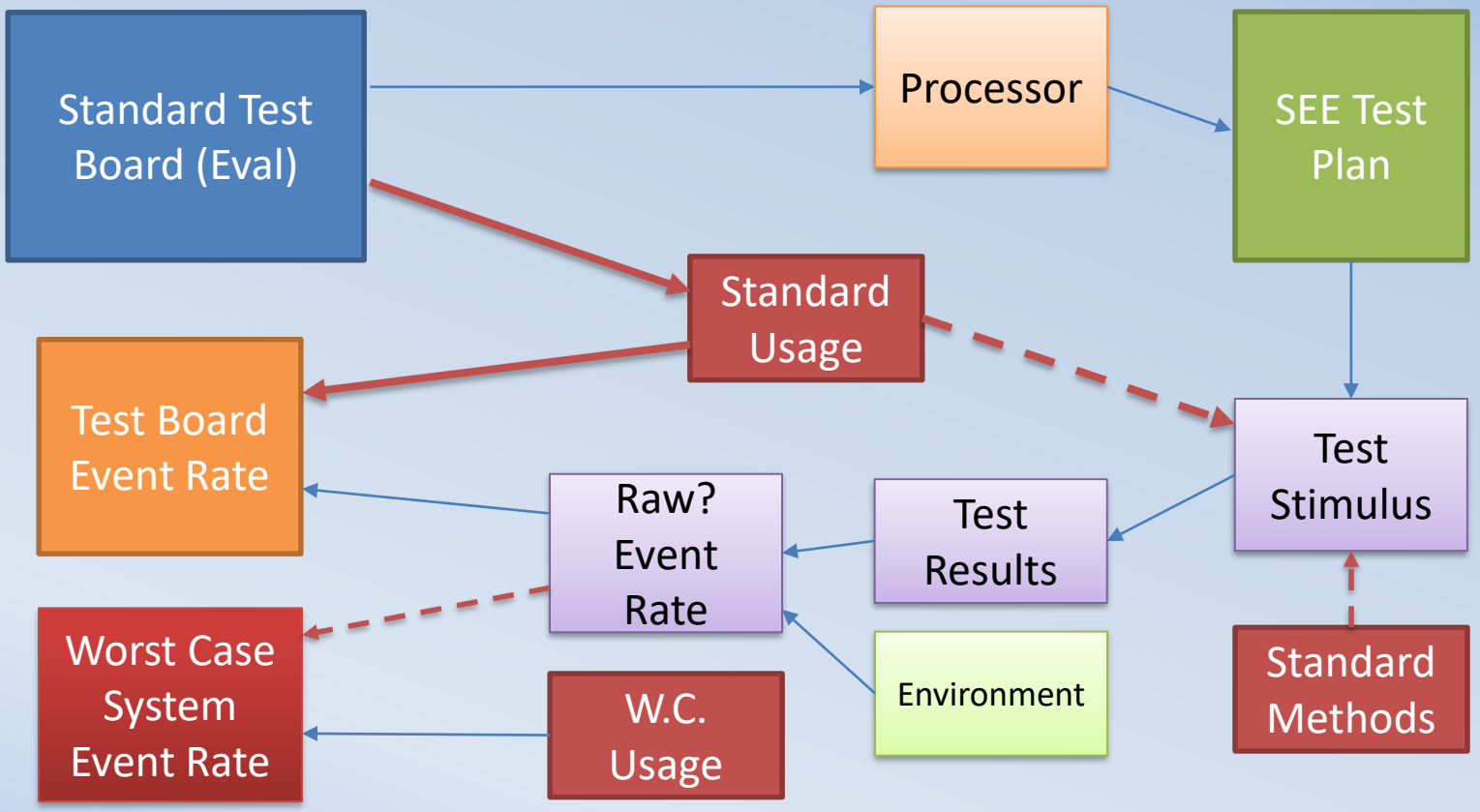
Old Approach



- Worst case usage in the old approach is just “everything matters all the time”
- Standard methods are well-known (but maybe not easy to implement)

National Aeronautics and Space Administration

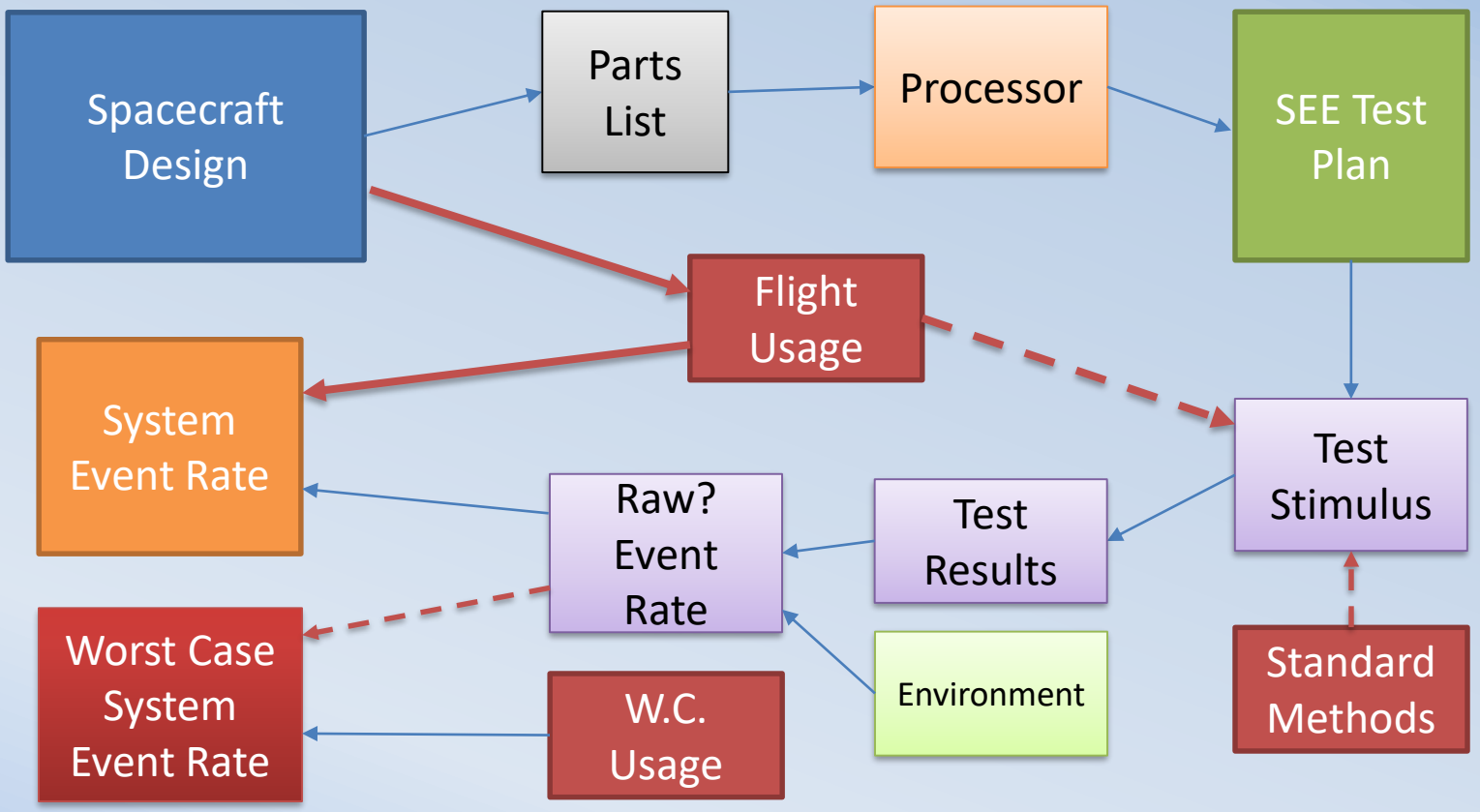
New Approach



- Standard methods can't be implemented, and only apply to 10% of device
- We get rates for the standard usage (not methods) of the board, not flight rates…
- Ultimately we have to translate the test board rate to the system rate – we can, perhaps go to a worst-case system rate...

National Aeronautics and Space Administration

The Problem(s)?



- Worst case usage getting very hard to identify
- Flight usage info not available, and does not contribute to generally useful worst-case rate info
- Raw-element/standard methods are not working on new hardware

Moving Forward

- Low-level structure testing has been morphing
  - Intel processors (collaboration with GSFC and Navy Crane)
    - Running register and cache tests
      - No events seen or
      - Machine check exceptions that can't be normalized or
      - We just get crashes
    - System-level tests – with Windows
      - Same set of events as above, we do get lots of "core dump" info – but sifting through what each event means is strange
      - Can run stress tests, look at different operating conditions
  - PowerPC
    - These continue to have very good documentation, but system-level worst-case is harder to get at
  - ARM
    - Very wide variety of ARM devices… some have excellent documentation, others have almost no documentation (for the SOC)

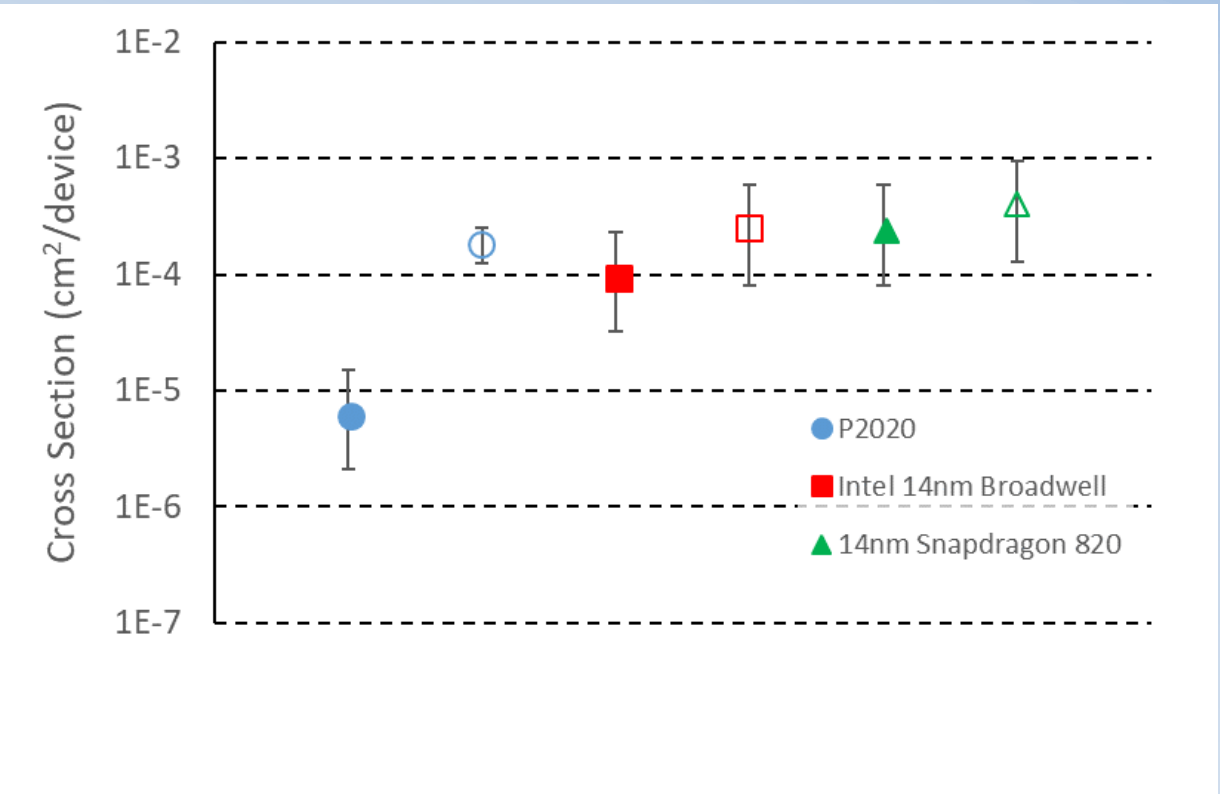
National Aeronautics
and Space Administration

Moving Forward

- System-level testing
  - Although a stop-gap, we are now recommending similar-to flight system testing to avoid missing something critical
  - The relationship between beam structure and test software operation (operating system and stress tests) is very tricky
    - Accelerated beam rates may easily create crashes where a report and retry was possible
  - What is the right thing to test with, assuming flight code is not available.
    - Even if flight code is available – is it the best to use, given that the flight code will probably change again? Also, specific flight code may emphasize something ore be really poor for maximizing value from beamtime

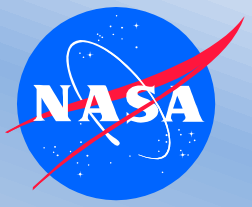
National Aeronautics
and Space Administration

Open vs. Closed Info



- Cross section for crashes while running “other” tests.
- Closed symbols are “low utilization”, while open symbols are “high utilization”.
- The P2020, although an SOC is easily configured to run in a very minimal mode. The Intel and Snapdragon devices are more realistic for new and future devices with minimal documentation.

Subsystem Testing



- How do you test the Ethernet controller, e.g.?
- The manufacturer has ways to evaluate and operate these with debug modules
  - Those may be hiding behind NDAs or worse (manufacturer may prefer to provide support because documentation and tools are too hard to use).
  - Even if you can test this way, is it really how the device will work in a real system?
- If testing as part of a system, you get system errors
  - How do you determine which system errors are due to the guilty subsystem, etc.
  - Do you, perhaps, focus on system errors rather than care about errors isolated to the subsystem?

Conclusion

- "Good" SEE data on processors increasingly difficult to get – traditional assurance methods no longer working
  - Software – putting custom low-level software in beam getting hard
  - Configuring board/SOC elements is very difficult without high-level languages, multithreading, etc.
  - Actual device operation controlled by multicore hypervisor, memory management, interrupt handling
- Worst case testing/rate evaluation is no longer viable
  - Nobody is going to use everything, and if they are they won't use it all 100% of the time

Conclusion

- Targeted testing of low-level stuff vs. system level
  - Almost doesn't matter, crash rates and machine checks dominate, and change maybe 3x
- Testing all subsystems or establishing a complete system-level test is hard, and likely requires a combination of test applications running…
- The actual test methods are in flux, so for now:
  - We are working on a combination of low-level tests and full system-level tests
  - The system-level tests are being carried out with semi-canned stress tests, but these are for reliability and throughput tests
  - Need to establish flux levels, how to handle mitigation (which the devices already have), etc.